Project 6

Matthew Nutt

December 16, 2025

Given a linear system $\mathbf{A}$ and a target vector $\vec{\mathbf{b}}$, my linear solver finds the vector $\vec{\mathbf{x}}$ that solves the linear equation $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{b}}$. First the QR decomposition of $\mathbf{A}$ provides $\mathbf{A} = \mathbf{QR}$, such that $\mathbf{Q}$ is an orthonormal matrix and $\mathbf{R}$ is a right-triangular matrix. Then $\mathbf{QR}\vec{\mathbf{x}} = \vec{\mathbf{b}}$, and since $\mathbf{Q}$ is orthonormal, $\mathbf{R}\vec{\mathbf{x}} = \mathbf{Q^T}\vec{\mathbf{b}}$. Thanks to the zeroes of $\mathbf{R}$, this equation can then be solved with backsubstitution.

Using modified modules from Projects 3, 4, and 5, my solver implements this algorithm in C++ for HLS onto an FPGA. As requested, I implemented two versions of my HLS design; one version, `solver_fixed.cpp`, inputs and outputs fixed-point values. The other version, `solver_float.cpp`, inputs and outputs floating-point values, though it still uses fixed point internally.

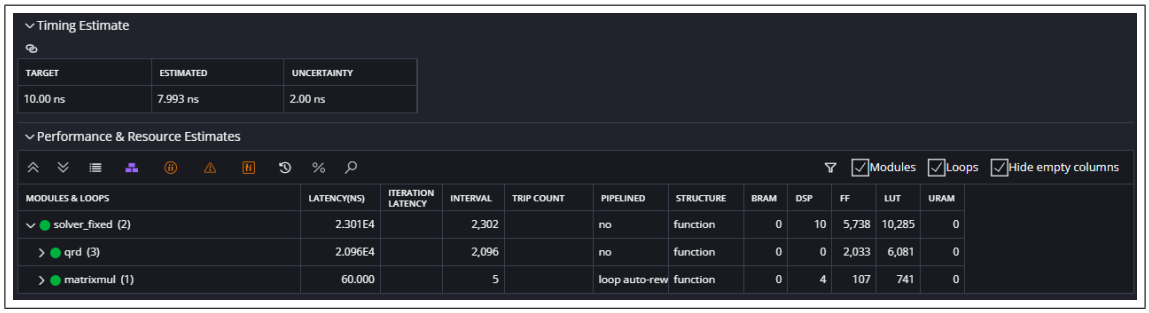
```
Running Test #1 ============================
A = [
[0.0999756, 0.0999756, 0.400024, 0.300049],
[0.199951, 0.800049, 0.599976, 0.5],
[0.900024, 0.0999756, 0.300049, 0.199951],
[0.300049, 0.0999756, 0.400024, 0.599976]
]
b = [0.199951, 0.0999756, 0.800049, 0.400024]
x = [0.80188, -0.37915, 0.29834, 0.128662]
b_reconstruct = [0.200195, 0.10022, 0.79895, 0.39917]
Difference = [-0.000244141, -0.000244141, 0.00109863, 0.000854492]
Running Test #2 ============================
A = [
[-0.0999756, -0.699951, 0.599976, 0.300049],
[-0.0999756, 0.400024, -0.699951, 0.5],
[-0.900024, 0.5, 0.400024, -0.900024],
[-0.5, 0.199951, 0.599976, -0.400024]
]
b = [-0.599976, -0.800049, 0.400024, 1]
x = [2.12024, 2.85449, 2.59485, 0.170532]
b_reconstruct = [-0.601929, -0.801025, 0.403442, 0.99939]
Difference = [0.00195313, 0.000976563, -0.00341797, 0.000610352]
Running Test #3 ============================
A = [
[-0.800049, -0.800049, 0.900024, 0],
[-0.599976, 0.199951, -0.800049, 0.699951],
[0.300049, 0.599976, -0.5, 0.699951],
[0.599976, 0.599976, 1, -0.0999756]
]
b = [-1, 0.400024, 1, -0.300049]
x = [1.05249, -0.442749, -0.56958, 0.950928]
b_reconstruct = [-1.00049, 0.401367, 1.00061, -0.298828]
Difference = [0.000488281, -0.00134277, -0.000610352, -0.0012207]
Tests Complete =============================
```

Test results for `solver_fixed.cpp`.

```
Running Test #1 ============================
A = [
[0.1, 0.1, 0.4, 0.3],
[0.2, 0.8, 0.6, 0.5],
[0.9, 0.1, 0.3, 0.2],
[0.3, 0.1, 0.4, 0.6]
]
b = [0.2, 0.1, 0.8, 0.4]
x = [0.80188, -0.37915, 0.29834, 0.128662]
b_reconstruct = [0.200208, 0.100391, 0.799011, 0.399182]
Difference = [-0.000207528, -0.000390626, 0.000988841, 0.000817865]
Running Test #2 ============================
A = [
[-0.1, -0.7, 0.6, 0.3],
[-0.1, 0.4, -0.7, 0.5],
[-0.9, 0.5, 0.4, -0.9],
[-0.5, 0.2, 0.6, -0.4]
]
b = [-0.6, -0.8, 0.4, 1]
x = [2.12024, 2.85449, 2.59485, 0.170532]
b_reconstruct = [-0.602099, -0.801355, 0.403491, 0.999475]
Difference = [0.00209945, 0.00135481, -0.00349125, 0.000524879]
Running Test #3 ============================
A = [
[-0.8, -0.8, 0.9, 0],
[-0.6, 0.2, -0.8, 0.7],
[0.3, 0.6, -0.5, 0.7],
[0.6, 0.6, 1, -0.1]
]
b = [-1, 0.4, 1, -0.3]
x = [1.05249, -0.442749, -0.56958, 0.950928]
b_reconstruct = [-1.00042, 0.40127, 1.00054, -0.298828]
Difference = [0.000415087, -0.00126952, -0.000537157, -0.00117189]
Tests Complete =============================
```

Test results for `solver_float.cpp`.

Using standard array partitioning pragmas in the solver function, and incorporating the existing pragmas of the other modules, the design was optimized for performance and packaged for synthesis. Vitis reports a 23 µs latency estimation on both designs.



Timing and resource utilization post-synthesis for `solver_fixed.cpp`.

Timing Estimate

TARGET	ESTIMATED	UNCERTAINTY
10.00 ns	7.993 ns	2.00 ns

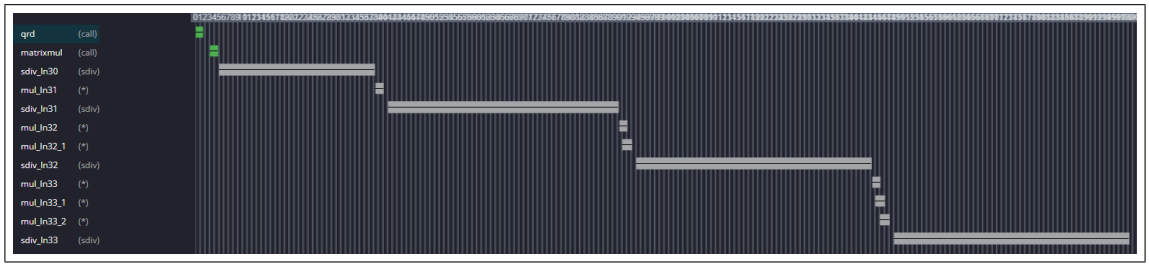
Performance & Resource Estimates

Modules Loops Hide empty columns

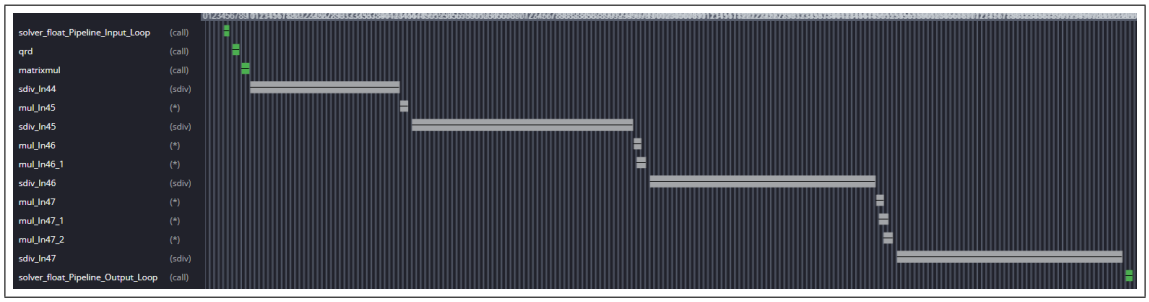
MODULES & LOOPS	LATENCY(NS)	ITERATION LATENCY	INTERVAL	TRIP COUNT	PIPELINED	STRUCTURE	BRAM	DSP	FF	LUT	URAM
solver_float (4)	2.321E4		2,322		no	function	0	10	7,552	19,001	0
solver_float_Pipeline_Input_Loop (1)	70.000		5		loop auto-rew	function	0	0	445	7,844	0
qrd (3)	2.096E4		2,096		no	function	0	0	2,028	6,150	0
matrixmul (1)	60.000		5		loop auto-rew	function	0	4	107	741	0
solver_float_Pipeline_Output_Loop (1)	60.000		5		loop auto-rew	function	0	0	5	365	0

Timing and resource utilization post-synthesis for `solver_float.cpp`.

The schedule view of each design indicates that division operations take up most of the clock cycles during computation.

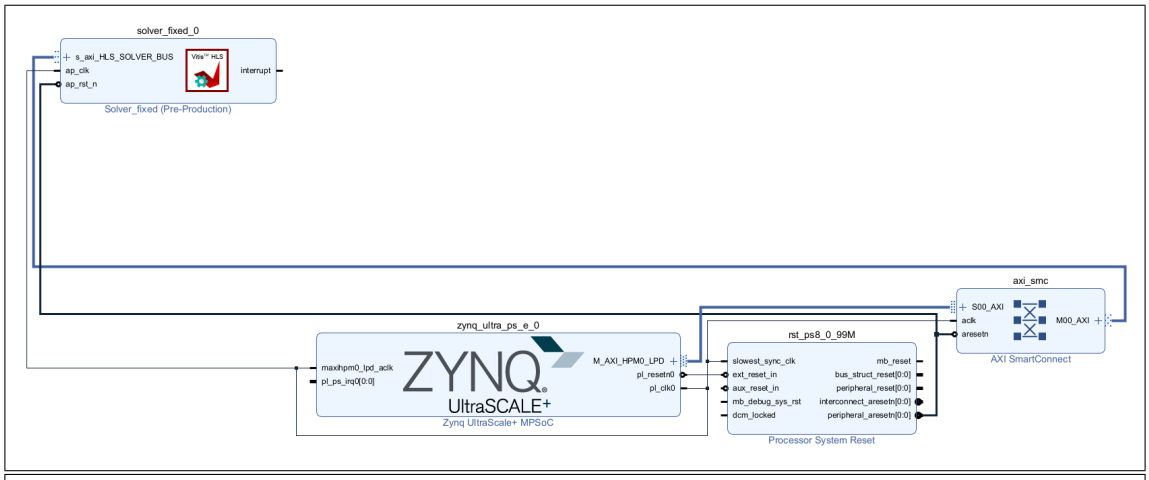


Timing schedule for `solver_fixed.cpp`.



Timing schedule for `solver_float.cpp`.

In Vivado, a block diagram of each design was assembled, and the final synthesis/implementation was completed.

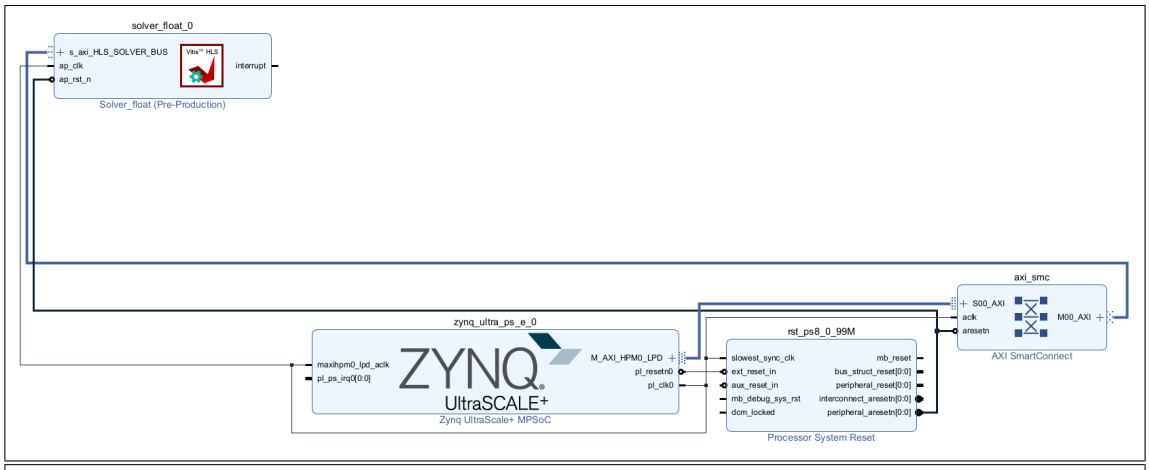


Resource	Utilization	Available	Utilization %
LUT	5391	70560	7.64
LUTRAM	9	28800	0.03
FF	4173	141120	2.96
BRAM	4	216	1.85
DSP	13	360	3.61

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.719 ns	Worst Hold Slack (WHS):	0.014 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9455	Total Number of Endpoints:	9455	Total Number of Endpoints:	4206

All user specified timing constraints are met.

Results from the Vivado implementation process of `solver_fixed.cpp`.



Resource	Utilization	Available	Utilization %
LUT	8706	70560	12.34
LUTRAM	9	28800	0.03
FF	5986	141120	4.24
BRAM	4	216	1.85
DSP	13	360	3.61

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.667 ns	Worst Hold Slack (WHS):	0.012 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	13158	Total Number of Endpoints:	13158	Total Number of Endpoints:	6017

All user specified timing constraints are met.

Results from the Vivado implementation process of `solver_float.cpp`.

Taking the resulting hardware platform into Vitis, two application files were created, `main_fixed.cpp` and `main_float.cpp`, built for their corresponding design version. Both applications verify the functionality of the HLS design on the PL of the devboard with three different linear systems. They also include software implementations of the same linear solver algorithm used in HLS, albeit substituting trigonometric functions in place of a CORDIC module for vector rotations.

```
Running Test #1 ================================
A = [
[0.0999756, 0.0999756, 0.400024, 0.300049],
[0.199951, 0.800049, 0.599976, 0.5],
[0.900024, 0.0999756, 0.300049, 0.199951],
[0.300049, 0.0999756, 0.400024, 0.599976]
]
b = [0.199951, 0.0999756, 0.800049, 0.400024]
HW Result --------------------
x = [0.80188, -0.37915, 0.29834, 0.128662]
Total time steps for hardware = 877
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 2.631e-05
SW Result --------------------
x = [0.803345, -0.379272, 0.295898, 0.130615]
Total time steps for software = 39428
Cycle counts per second = 3.33333e+07
Time in seconds for software = 0.00118284
=== Speedup of hardware vs software = 44.9578
Running Test #2 ================================
A = [
[-0.0999756, -0.699951, 0.599976, 0.300049],
[-0.0999756, 0.400024, -0.699951, 0.5],
[-0.900024, 0.5, 0.400024, -0.900024],
[-0.5, 0.199951, 0.599976, -0.400024]
]
b = [-0.599976, -0.800049, 0.400024, 1]
HW Result --------------------
x = [2.12024, 2.85449, 2.59485, 0.170532]
Total time steps for hardware = 849
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 2.547e-05
SW Result --------------------
x = [2.12256, 2.8595, 2.60107, 0.177734]
Total time steps for software = 37539
Cycle counts per second = 3.33333e+07
Time in seconds for software = 0.00112617
=== Speedup of hardware vs software = 44.2155
Running Test #3 ================================
A = [
[-0.800049, -0.800049, 0.900024, 0],
[-0.599976, 0.199951, -0.800049, 0.699951],
[0.300049, 0.599976, -0.5, 0.699951],
[0.599976, 0.599976, 1, -0.0999756]
]
b = [-1, 0.400024, 1, -0.300049]
HW Result --------------------
x = [1.05249, -0.442749, -0.56958, 0.950928]
Total time steps for hardware = 850
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 2.55e-05
SW Result --------------------
x = [1.05591, -0.447266, -0.570068, 0.952515]
Total time steps for software = 37563
Cycle counts per second = 3.33333e+07
Time in seconds for software = 0.00112689
=== Speedup of hardware vs software = 44.1918
Tests Complete ================================
```

Serial monitor output for all three linear systems tested on the fixed-point version of the project.

```
Running Test #1 ================================
A = [
[0.1, 0.1, 0.4, 0.3],
[0.2, 0.8, 0.6, 0.5],
[0.9, 0.1, 0.3, 0.2],
[0.3, 0.1, 0.4, 0.6]
]
b = [0.2, 0.1, 0.8, 0.4]
HW Result --------------------
x = [0.80188, -0.37915, 0.29834, 0.128662]
Total time steps for hardware = 1001
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 3.003e-05
SW Result --------------------
x = [0.803354, -0.379573, 0.295731, 0.131098]
Total time steps for software = 380
Cycle counts per second = 3.33333e+07
Time in seconds for software = 1.14e-05
=== Speedup of hardware vs software = 0.37962
Running Test #2 ================================
A = [
[-0.1, -0.7, 0.6, 0.3],
[-0.1, 0.4, -0.7, 0.5],
[-0.9, 0.5, 0.4, -0.9],
[-0.5, 0.2, 0.6, -0.4]
]
b = [-0.6, -0.8, 0.4, 1]
HW Result --------------------
x = [2.12024, 2.85449, 2.59485, 0.170532]
Total time steps for hardware = 958
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 2.874e-05
SW Result --------------------
x = [2.12198, 2.8591, 2.60042, 0.177707]
Total time steps for software = 232
Cycle counts per second = 3.33333e+07
Time in seconds for software = 6.96e-06
=== Speedup of hardware vs software = 0.242171
Running Test #3 ================================
A = [
[-0.8, -0.8, 0.9, 0],
[-0.6, 0.2, -0.8, 0.7],
[0.3, 0.6, -0.5, 0.7],
[0.6, 0.6, 1, -0.1]
]
b = [-1, 0.4, 1, -0.3]
HW Result --------------------
x = [1.05249, -0.442749, -0.56958, 0.950928]
Total time steps for hardware = 956
Cycle counts per second = 3.33333e+07
Time in seconds for hardware = 2.868e-05
SW Result --------------------
x = [1.05506, -0.446356, -0.57004, 0.951822]
Total time steps for software = 232
Cycle counts per second = 3.33333e+07
Time in seconds for software = 6.96e-06
=== Speedup of hardware vs software = 0.242678
Tests Complete ================================
```

Serial monitor output for all three linear systems tested on the floating-point version of the project.

As the results indicate, the fixed-point version of the software suffers tremendous delays compared to its matching PL. This is likely due to the lack of optimization for the fixed-point datatype; the PL is synthesized to directly process the datatype, but the software likely performs inefficient conversions in order to carry out operations. One remedy I learned from another student was to implement fixed-point versions of common operations like multiplication and division, but out of the box, the software performs very poorly with fixed-point.

In stark contrast, the floating-point version of the software is several orders of magnitude faster, even outpacing the PL. With native operations in floating point, as well as access to libraries such as `cmath`, this version suits the software much better than fixed-point. The PL also suffered some additional delay compared to the pure fixed-point version, since it first has to convert between the two datatypes before it can complete its algorithm.

Test #	**Fixed-Point HW**	**Fixed-Point SW**	**Floating-Point HW**	**Floating-Point SW**
#1	877 cycles	39428 cycles	1001 cycles	380 cycles
#2	849 cycles	37539 cycles	958 cycles	232 cycles
#3	850 cycles	37563 cycles	956 cycles	232 cycles
Total	2576 cycles	114530 cycles	2915 cycles	844 cycles